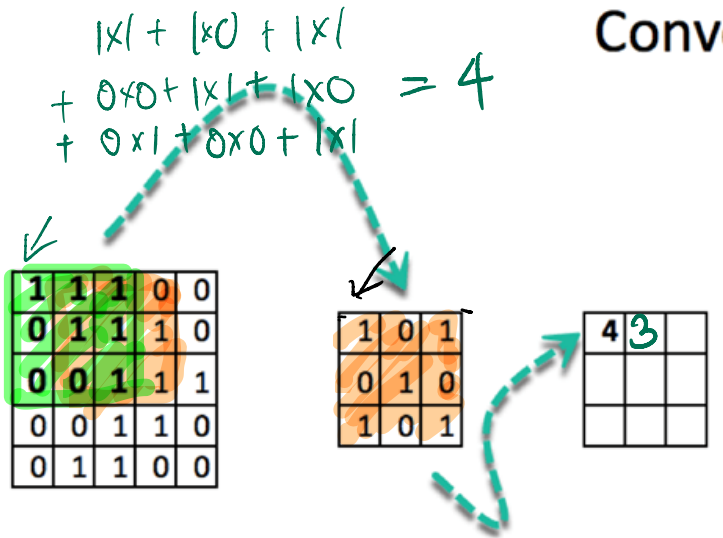
# Convolutional Neural Network

(CNN)

# Classification of images

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 171 | 183 | 181 | 171 | 165 | 167 | 164 | 148 | 150 | 154 | 161 | 163 | 160 | 173 | 176 | 182 | 184 | 179 | 180 | 182 | 184 |
| 166 | 159 | 149 | 144 | 146 | 148 | 142 | 129 | 120 | 115 | 127 | 138 | 150 | 152 | 163 | 178 | 189 | 183 | 187 | 192 | 198 |
| 155 | 134 | 124 | 130 | 136 | 141 | 130 | 113 | 101 | 107 | 117 | 133 | 141 | 153 | 159 | 179 | 189 | 184 | 183 | 191 | 200 |
| 134 | 114 | 105 | 114 | 122 | 124 | 107 | 95 | 98 | 103 | 120 | 129 | 144 | 161 | 174 | 183 | 176 | 177 | 181 | 190 | 202 |
| 128 | 106 | 91 | 94 | 99 | 92 | 79 | 76 | 86 | 96 | 117 | 130 | 145 | 161 | 167 | 161 | 152 | 171 | 190 | 200 | 213 |
| 113 | 93 | 75 | 71 | 73 | 76 | 72 | 70 | 63 | 72 | 98 | 133 | 144 | 128 | 134 | 144 | 157 | 171 | 186 | 193 | 205 |
| 86 | 79 | 66 | 62 | 65 | 72 | 78 | 74 | 68 | 75 | 95 | 126 | 121 | 117 | 124 | 155 | 189 | 189 | 199 | 205 | 212 |
| 72 | 75 | 68 | 63 | 66 | 74 | 74 | 62 | 69 | 75 | 94 | 111 | 108 | 117 | 130 | 166 | 198 | 193 | 204 | 210 | 211 |
| 67 | 70 | 67 | 65 | 68 | 68 | 62 | 60 | 70 | 78 | 89 | 96 | 103 | 127 | 150 | 181 | 191 | 190 | 190 | 192 | 197 |
| 69 | 69 | 73 | 79 | 85 | 85 | 73 | 79 | 75 | 80 | 86 | 97 | 123 | 148 | 165 | 175 | 169 | 189 | 186 | 183 | 190 |
| 76 | 72 | 80 | 86 | 90 | 90 | 86 | 89 | 88 | 87 | 87 | 100 | 108 | 110 | 118 | 125 | 142 | 164 | 170 | 173 | 186 |
| 84 | 81 | 86 | 87 | 88 | 84 | 78 | 88 | 80 | 76 | 76 | 93 | 91 | 81 | 83 | 97 | 126 | 140 | 148 | 157 | 171 |
| 88 | 87 | 92 | 91 | 88 | 83 | 78 | 88 | 74 | 70 | 74 | 95 | 85 | 70 | 68 | 89 | 129 | 135 | 142 | 150 | 163 |
| 85 | 85 | 89 | 87 | 84 | 80 | 78 | 85 | 74 | 74 | 81 | 94 | 84 | 72 | 73 | 99 | 139 | 148 | 152 | 156 | 164 |
| 97 | 90 | 88 | 80 | 75 | 78 | 83 | 89 | 93 | 101 | 110 | 113 | 110 | 111 | 118 | 139 | 166 | 181 | 183 | 183 | 185 |
| 130 | 115 | 104 | 91 | 84 | 80 | 86 | 87 | 97 | 108 | 119 | 121 | 127 | 134 | 143 | 155 | 168 | 185 | 191 | 192 | 194 |
| 162 | 147 | 132 | 116 | 107 | 103 | 101 | 98 | 99 | 100 | 108 | 117 | 128 | 127 | 133 | 141 | 156 | 170 | 179 | 185 | 193 |
| 176 | 162 | 148 | 131 | 124 | 115 | 109 | 110 | 106 | 103 | 108 | 126 | 134 | 135 | 138 | 143 | 163 | 166 | 176 | 181 | 192 |
| 187 | 175 | 162 | 152 | 144 | 131 | 124 | 125 | 119 | 117 | 120 | 127 | 147 | 154 | 160 | 172 | 184 | 176 | 180 | 186 | 192 |
| 189 | 178 | 166 | 157 | 150 | 137 | 122 | 120 | 121 | 122 | 120 | 122 | 139 | 168 | 184 | 203 | 210 | 203 | 203 | 202 | 200 |
| 189 | 178 | 169 | 161 | 155 | 144 | 127 | 121 | 124 | 128 | 128 | 134 | 155 | 175 | 192 | 208 | 213 | 217 | 216 | 213 | 206 |

# Filters


1x1 + 1x0 + 1x1
+ 0x0 + 1x1 + 1x0 = 4
+ 0x1 + 0x0 + 1x1
Convolution
1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
1 0 1
0 1 0
1 0 1
4 3

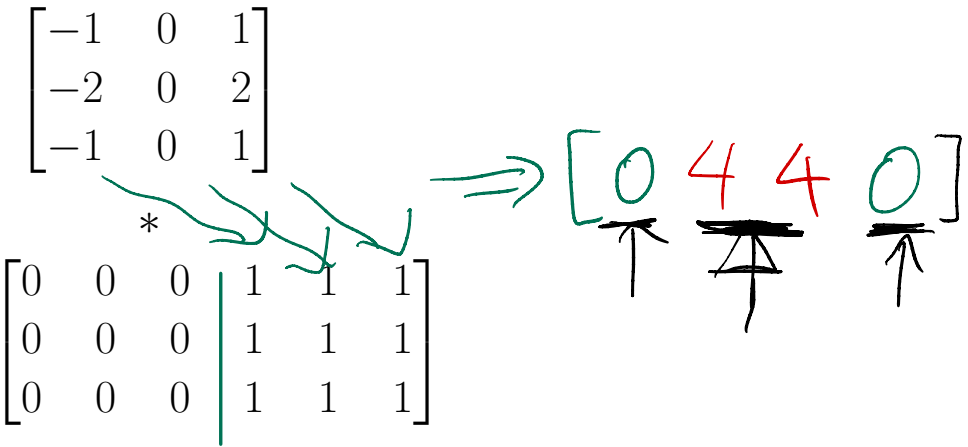
# Vertical edge detection

$\longrightarrow$

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$\longrightarrow$

# Vertical edge detection

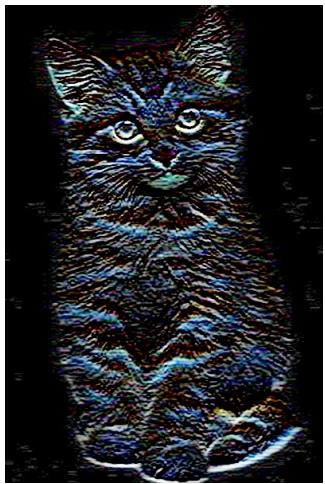


# Horizontal edge detection

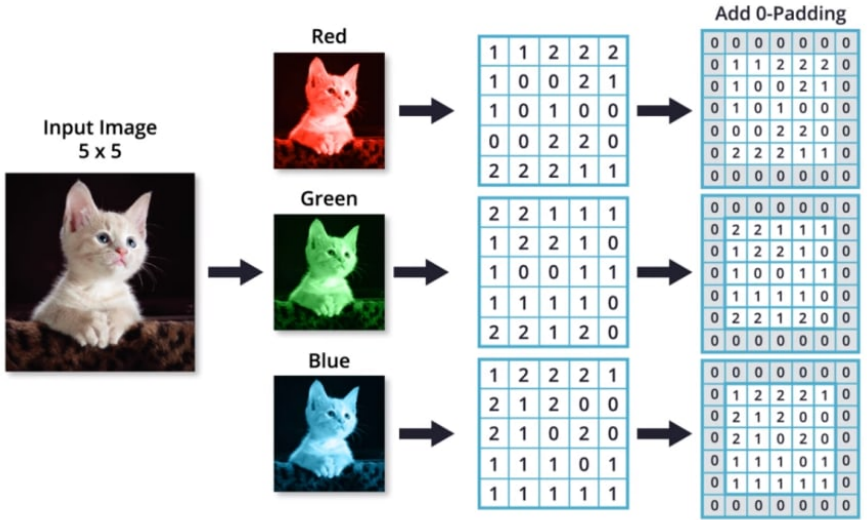
$\longrightarrow$

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

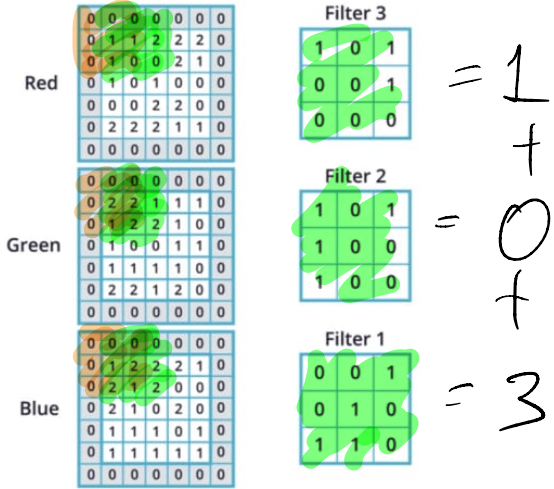
$\longrightarrow$

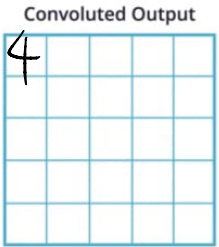


# RGB images



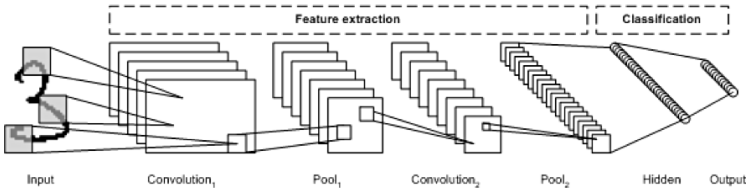
# RGB filters


Red
Green
Blue
Filter 3
Filter 2
Filter 1
= 1
+
= 0
+
= 3


Convoluted Output
4

# ZIP Code reading

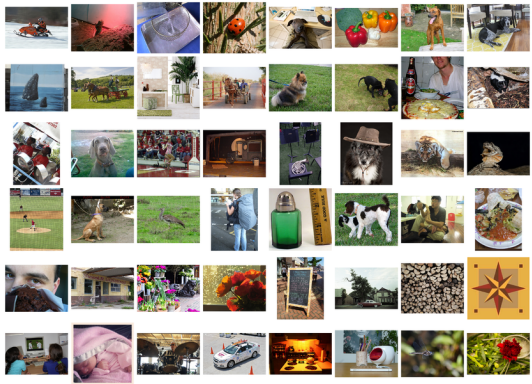
Yann Lecun




LeNet

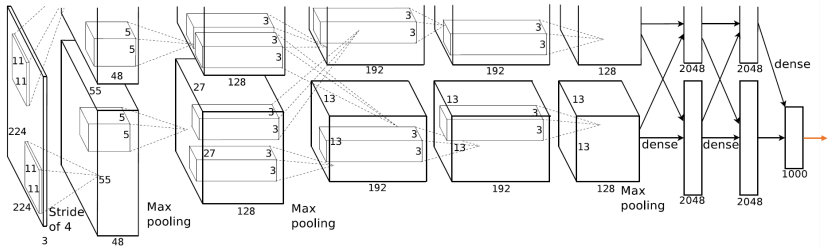
- Lecun et al. (1989) used Convolutional Neural Network - 95% accuracy

# Imagenet challenge (2012)



- 1.2 labeled training images: 1000+ images in each of 1000 categories
- 150,000 nonlabeled testing images

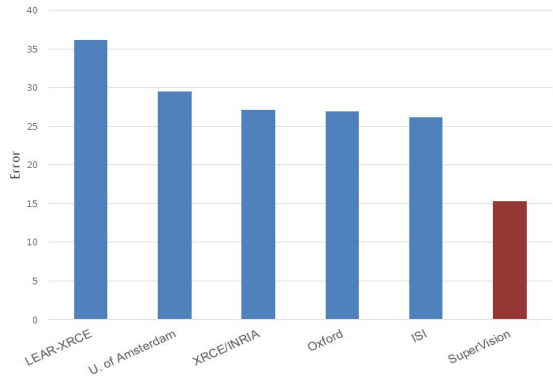
# AlexNet




- Geoff Hinton, Alex Krizhevsky and Ilya Sutskever, under the name of *SuperVision*, trained ImageNet with an 8-layer CNN which has 60 million parameters.
- These parameters could be trained with the help of two GTX 580 GPUs.

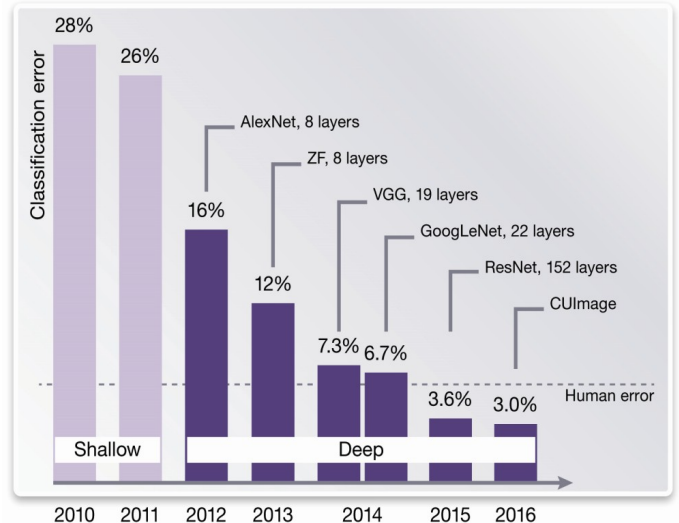
# ImageNet 1K Competition

(Fall 2012)




- SuperVision won the competition with 15.3% test error rate compared to 26.2% achieved by the second-best entry.

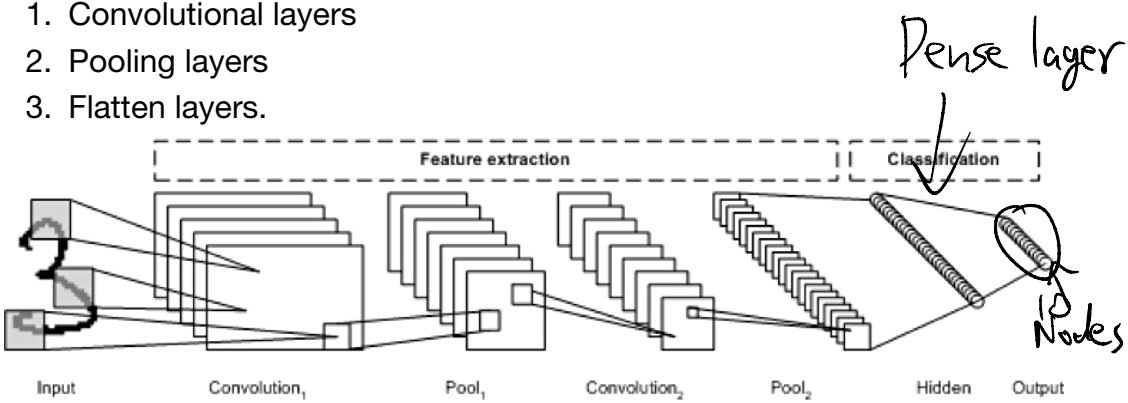
# Imagenet results (2010-2016)



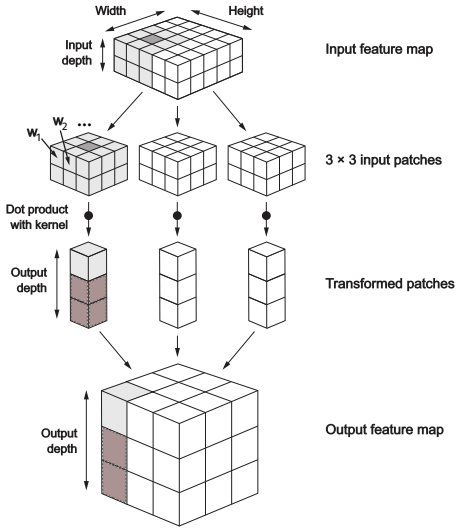
# Main layers

There are three main layers in a CNN:

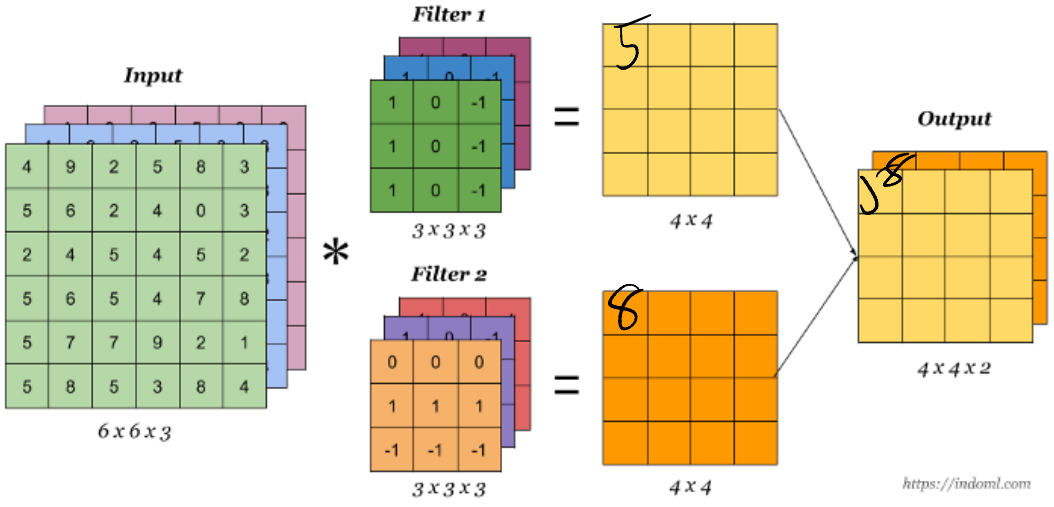
1. Convolutional layers
2. Pooling layers
3. Flatten layers.



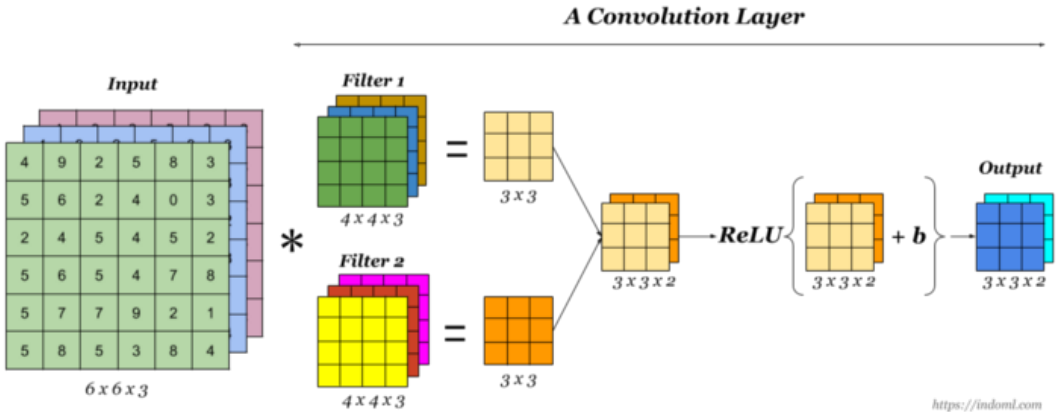
# Convolution layers



# Multiple filters



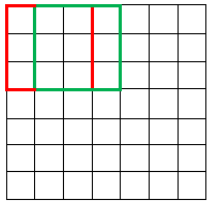
# A simple CNN


A Convolution Layer
Input
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
6 x 6 x 3
*
Filter 1
4 x 4 x 3
=
3 x 3
Filter 2
4 x 4 x 3
=
3 x 3
3 x 3 x 2
ReLU
3 x 3 x 2
+ b
Output
3 x 3 x 2
https://indoml.com

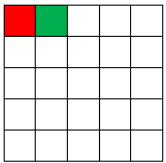
# Stride

1x1 Stride.

7 x 7 Input Volume

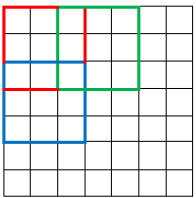


5 x 5 Output Volume

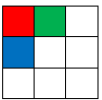


2x2 stride.

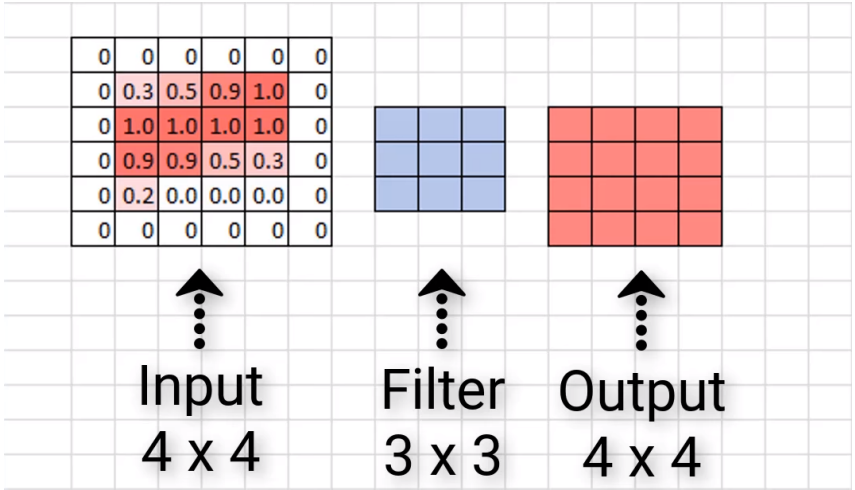
7 x 7 Input Volume



3 x 3 Output Volume



# Padding




- In case you want the output to be bigger

# Computing the dimension of output

**Example:**

- Input $227 \times 227 \times 3$
- 96 Filters of size $11 \times 11 \times 3$
- Stride $= 2$
- No padding

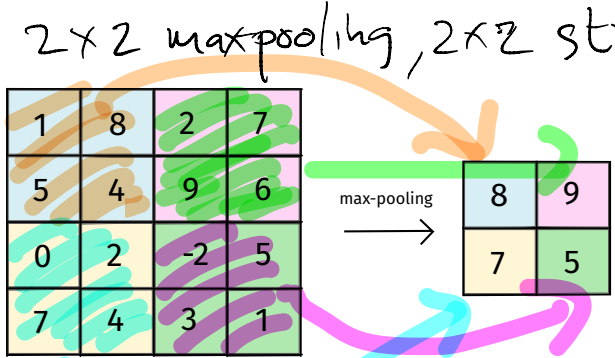
| Input | Filter | Output |
|---|---|---|
| $11 \times 11$ | $11 \times 11$ | $1 \times 1$ |
| $13 \times 13$ | $11 \times 11$ | $2 \times 2$ |

$227 \times 227 + 11 \times 11$

$$\rightsquigarrow \frac{227 - 11 + 2}{2} = 109 \times 109$$

96 filters $\Rightarrow 109 \times 109 \times 96$

# Max-poolings



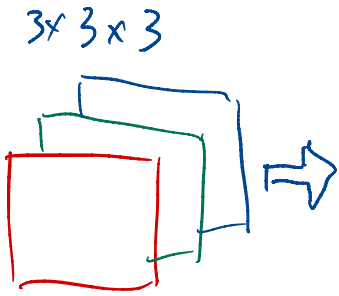

- helps reduce dimensions ⇒ less layers to train

Avg-pooling

# Flatten layer

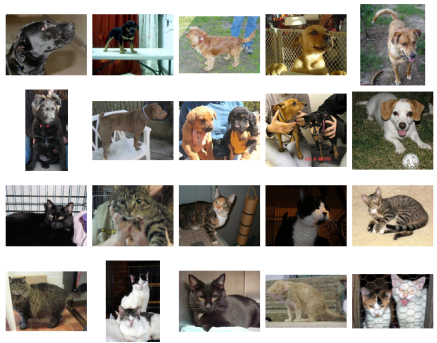
| | | |
|---|---|---|
| 1 | 1 | 0 |
| 4 | 2 | 1 |
| 0 | 2 | 1 |

| |
|---|
| 1 |
| 1 |
| 0 |
| 4 |
| 2 |
| 1 |
| 0 |
| 2 |
| 1 |

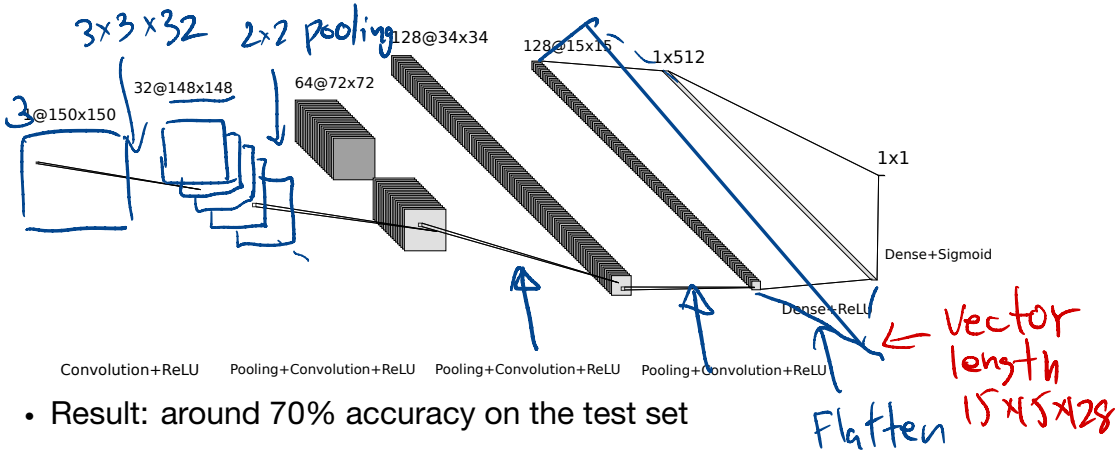


# Convolutional Neural Network: Example

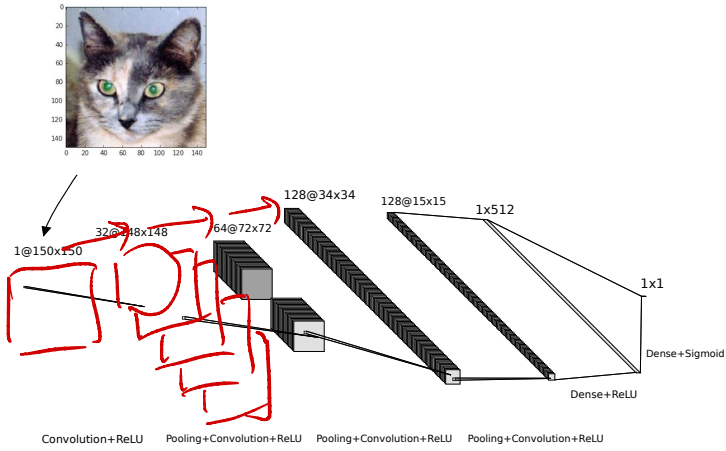
Dogs vs. Cats dataset (https://www.kaggle.com/c/dogs-vs-cats/data)



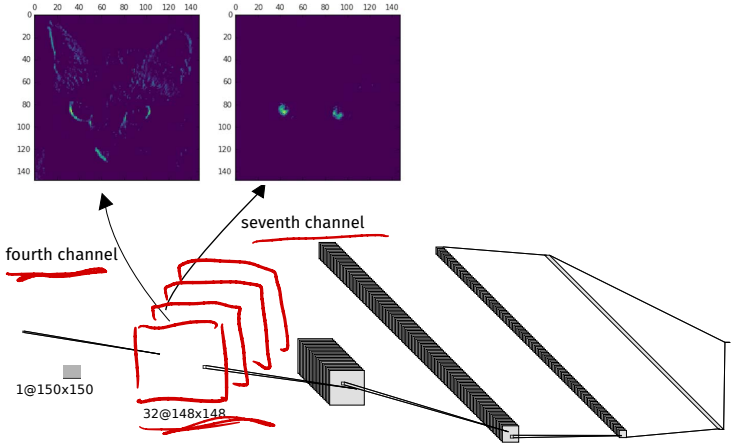
- Training images: 2000 images
- Testing images: 1000 images




- Result: around 70% accuracy on the test set
- Notice the ReLU activations after all hidden layers
- sigmoid at the end for binary classification

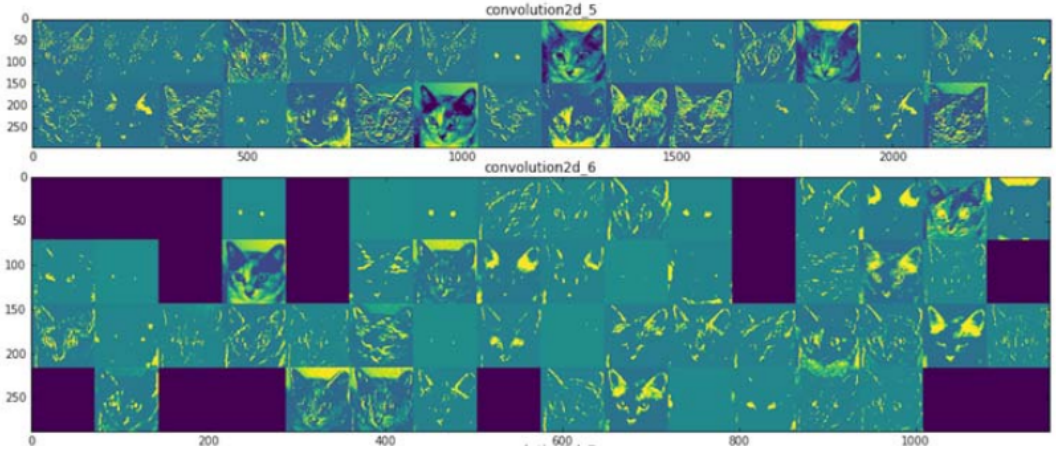



- Let's see what will happen to this image after the first convolutional layer.

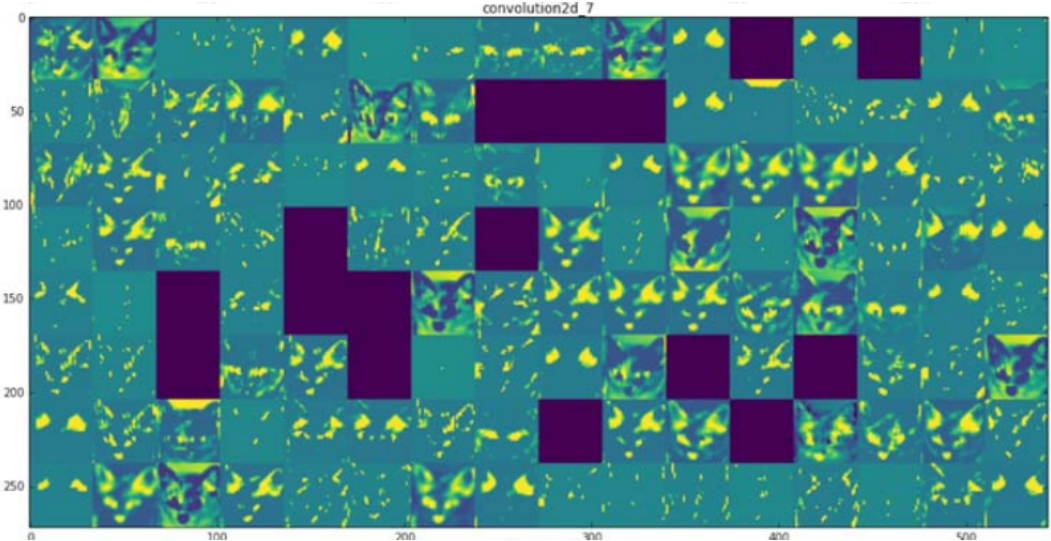



- It looks like the fourth layer detects diagonal edges and the seventh layer detects green pixels.

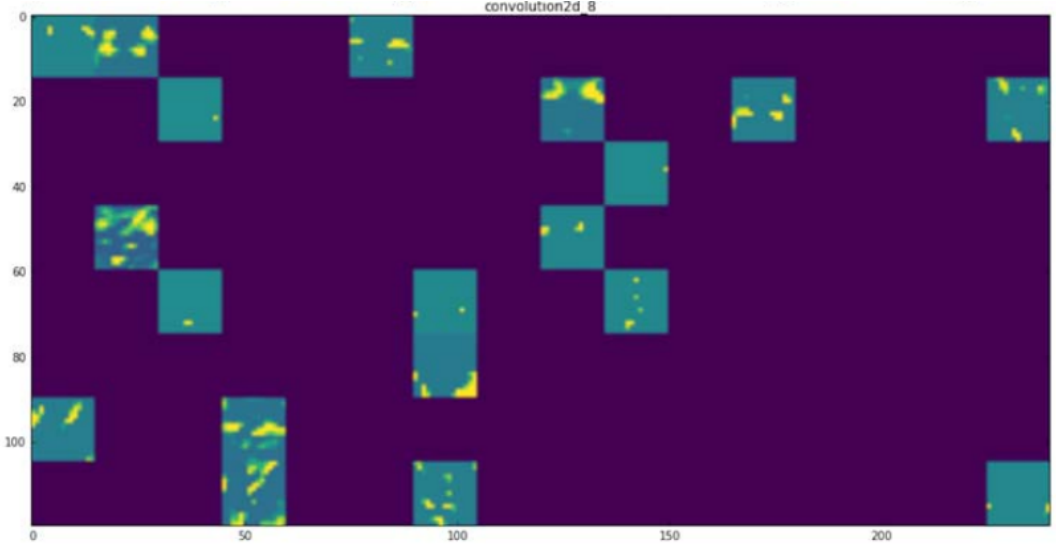
Filtered images from the first two convolutional layers:



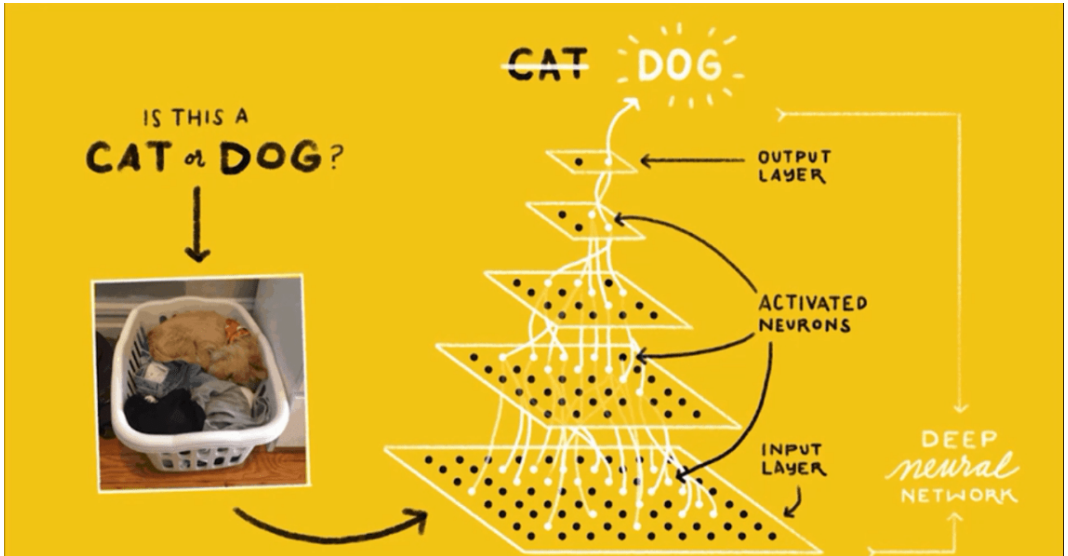
after the third layers:



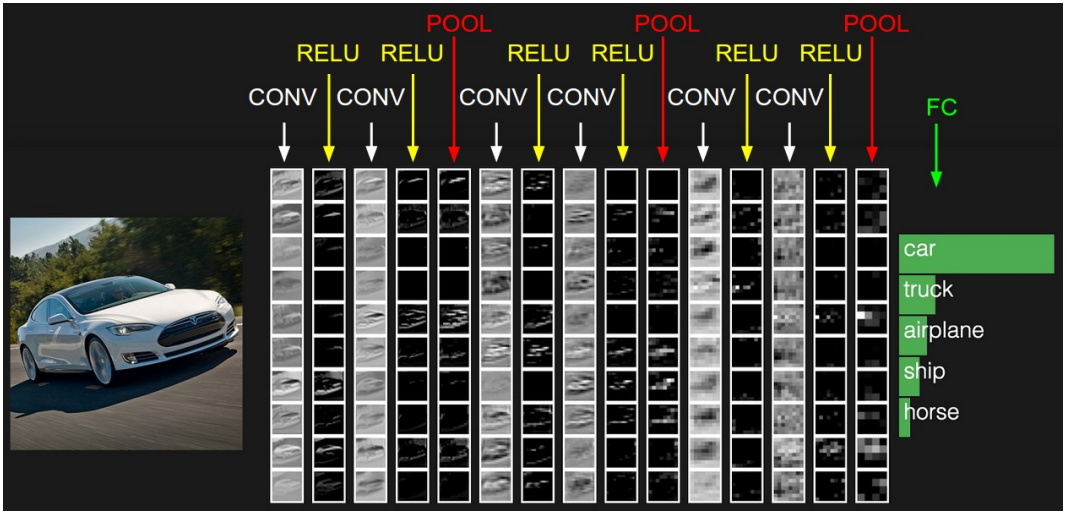
after the fourth layers:



# Another simplified picture of CNN



# Multiclass classification



Positives

- gives best performance on computer vision tasks
- incremental construction, meaning that one can add or modify existing layers to suit a specific task

Negatives

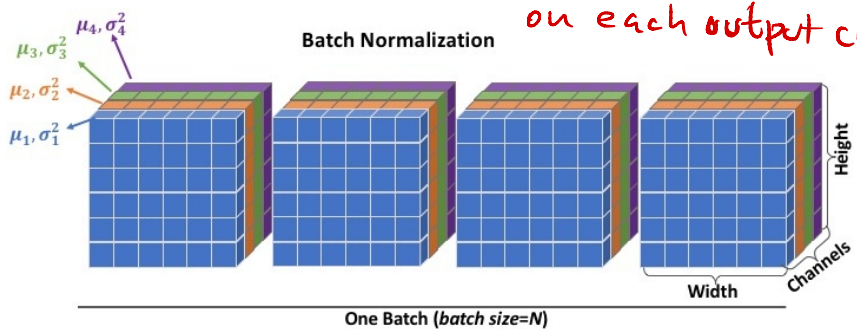
- computational intensive: training millions of parameters requires GPUs
- no unified theoretical support

# Other techniques

# Data augmentation

- Translation, rotation, flipping, cropping, color adjustment etc.
- More data to train $\rightarrow$ better predictions.

# Batch normalization (normalization on each output channel)

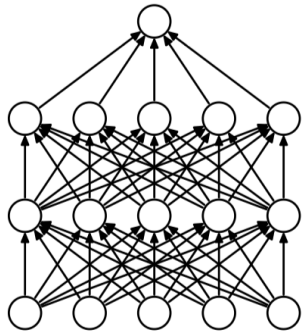



- Cells with the same color are subtracted by the same mean and divided by the same variance.
- Prevents the cell values from becoming too large. It also makes the training faster.

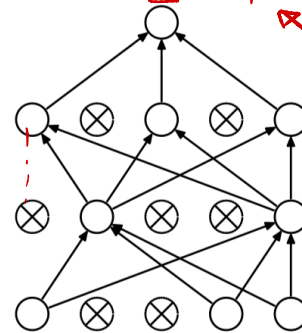
# Dropouts

User specifies $\underline{P}$

probability of the node to be "closed"



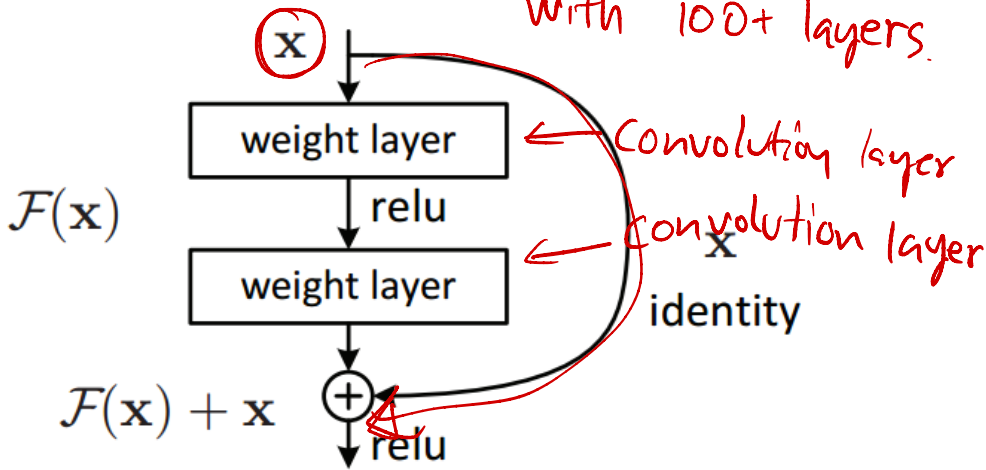
(a) Standard Neural Net



(b) After applying dropout.

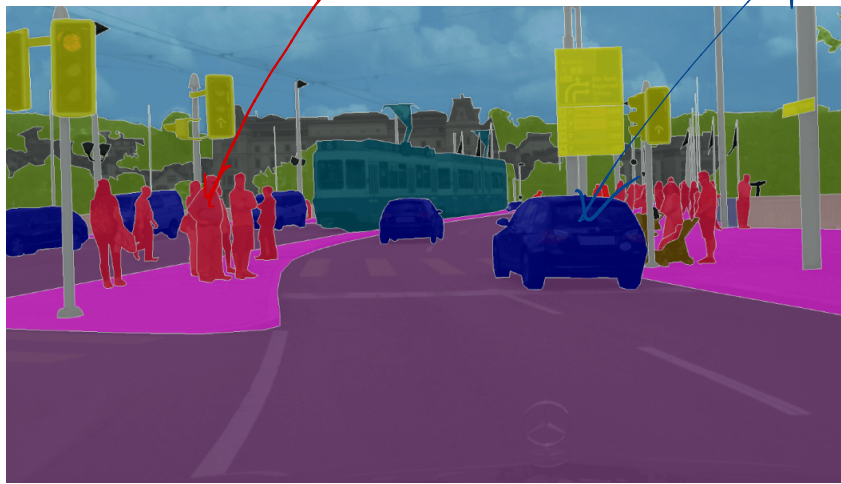
- Some of the nodes are randomly deactivated during each iteration.
- Prevents overfitting on training data. Improves model generalization.

# Skip connection



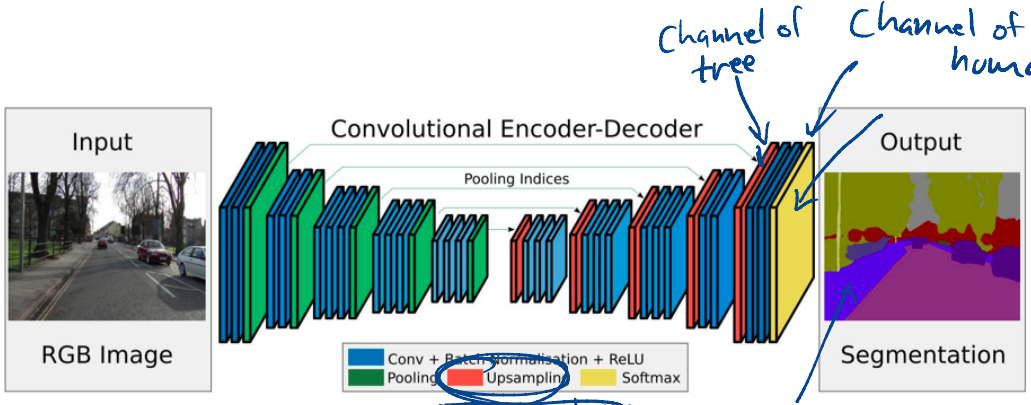

- Prevents the outputs of each layers from becoming zero.

# Segmentation




- Road segmentation for self-driving car.

# U-Net


Channel of tree
Channel of human
Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation